Context-aware Policy Matching in Event-driven Architecture

Shao-you Cheng
r93070@csie.ntu.edu.tw

Wan-rong Jih
jih@agents.csie.ntu.edu.tw

Jane Yung-jen Hsu
yjhsu@csie.ntu.edu.tw

Computer Science and Information Engineering,
National Taiwan University, Taiwan

Motivation

Applications for supporting pervasive computing and agility are the current trend in software development. The service-oriented architecture (SOA) enables connection between consumers and service providers in a loosely-coupled way to improve flexibility and extensibility. This architecture is static, which utilizes predefined sequences of actions and fixed policies. Ongoing processes cannot adapt to dynamic changes in the environmental conditions or *context*.

Imagine the situation where a real estate broker shows her client a house for sale, matching the preference profile provided by the potential buyer. While the buyer likes the general location of the house, he considers it unacceptable due to the unexpected traffic noise from a nearby street. Instead of proceeding with the original plan to show another house on the same street, an experienced broker should adjust the plan in light of this additional constraint. The broker connects to the multiple listing service with her mobile device and downloads a newly listed house within minutes of the current location that better satisfies the client's requirements.

Such a scenario can facilitate the introduction of *events* into SOA (He 2003). Real-time changes are modeled as events, which in turn trigger changes of states for the workflow to meet the business needs. In this scenario, events can be ”shows client a house”, ”add client's requirements”, etc.

Introduction

This research explores the role of context-aware policy matching in an *event-driven architecture* (EDA). In particular, a context-aware rule engine is adopted to derive conclusions based on the current contexts and business policies. Figure 1 shows the functional modules of the prototype designed to demonstrate the advantages of the proposed approach. In the *Underlying Architecture* layer, standard SOA is combined with EDA. The *Context-Aware Rule Engine* layer consists of three distinct agents for collecting preference profiles, ambient or context information, and dynamic events. All information collected will be forwarded to the Rule Engine, augmented with the Rule Repository and Context Ontology. Results from the Rule Engine will be given to the Action Agent that performs the desired sequence of actions. The following sections contain detailed description of each module.

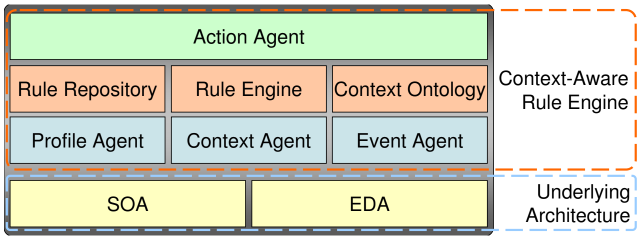

Figure 1: Context-aware policy matching framework

Event-driven Architecture

The SOA use a simple and clear interface to bind all participating software components and provides service reuse. However, it lacks proactive ability. Event notification is the core of EDA, it can tightly bind services, events, and consumers in a dynamic environment. Events are more likely to complement, not replace, services in SOA. Since EDA has been touted as the “the next big thing” on the horizon of software development methodology (Schulte 2004). It is generally believed that event-driven services under SOA gain benefits from the features of both architectures (Hanson 2005).

Conventional event-driven design is also referred to as message-based systems, where a message here is a form of an event. Specifications of WS-Notification family and WS-Eventing define the standard Web services approaches for event handling. When events occur, as shown in Figure 2, service producers publish the messages which will be delivered, e.g. via publish-and-subscribe, to the event consumers.

Context-Aware Rule Engine

The proposed framework utilizes the Jess rule engine (Friedman-Hill 2005) to provide inference results. Jess processes the rules and facts using the Rete algorithm (Forgy 1982). User preferences are represented as policy rules that are matched in the reasoning process.

Contexts refer to the various dynamic aspects of the environment, for example, location, time, and people (Dey

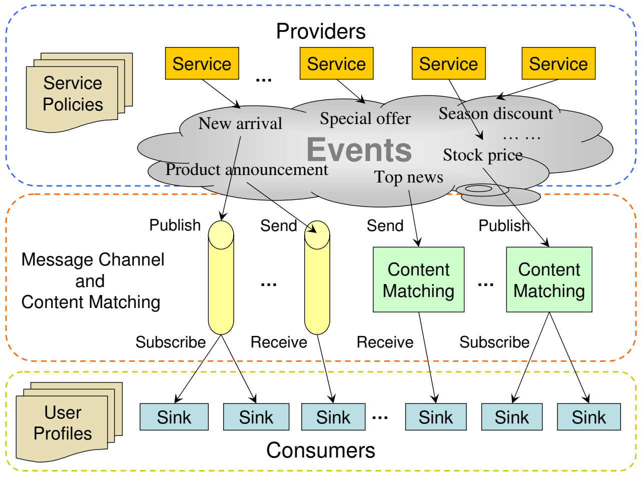

Figure 2: Message-based event-driven architecture

2001). Static rules cannot react to the dynamically changing contexts. As a result, context-aware rules are defined to use dynamic facts in the knowledge base, which may result in different conclusions depending on the current context. In our previous work, we successfully implemented context-aware rule-based reasoning on Java-enabled mobile devices, such as the iPAQ, to perform access control of sensitive information (Jih, Cheng, & Hsu 2005).

A context agent monitors and perceives any environmental changes. Figure 3 shows that context information may be captured by various multi-modal sensors, such as GPS or RFID. The context agent filters the contexts that have been detected, passing the selected contexts to the rule engine. Similarly, the event agent perceives and filters the relevant events of its surrounding area, and the profile agent keeps track of the user's preferences.

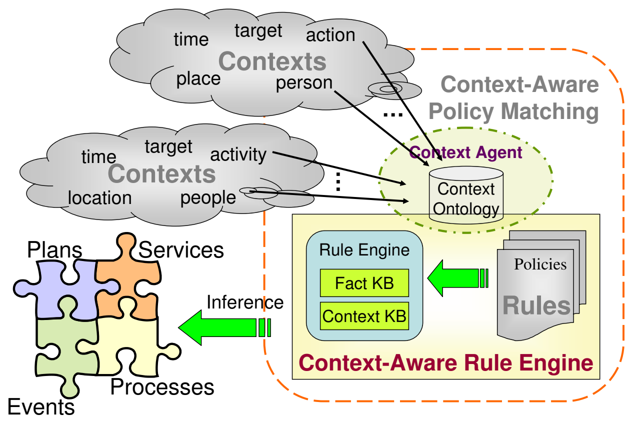

Figure 3: Context-aware rule engine

Now, let's examine the scenario of the real estate broker again. The buyer's rejection of the current house generates two events. First, the buyer is not satisfied with the choice, so he needs additional recommendations. Second, the buyer preference profile is incomplete, and should be modified to reflect his preference for a quiet location. The former event triggers the broker's mobile device to request for additional listing from the server, subject to successful matching between the updated preference profile and updated new listing.

When contexts are encoded in the rule sets, inconsistency in the terminology is not only confusing, but also leads to incorrect reasoning outcomes. Given that most useful contexts are tightly related to the problem domain (e.g. real estate), *ontology* can be used to bridge the semantic gap among different vocabularies. For example, while *location* is a common component of context defined in many applications, sometimes it is referred to as *place*, *space*, or *area*, and so on. A context ontology helps generate a complete model of the different contexts (H.Wang *et al.* 2004) for a given domain. A specific context might directly derive from a more generic one, aggregate to a complex context, or up to an abstract context. Moreover, the context ontology will support hierarchical views of contexts (Gu *et al.* 2004) to improve the reasoning power of the rule engine.

Conclusion

This paper describes the design of a proposed framework for deploying a context-aware rule engine to the event-driven services platform in order to provide agile and real-time services. The design uses an agent architecture and a rule engine for flexibility and scalability in software development. A context ontology is utilized to resolve inconsistent vocabularies in knowledge sharing and rule merging.

References

Dey, A. K. 2001. Understanding and using context. *Personal Ubiquitous Computing* 5(1):4–7.

Forgy, C. 1982. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence* 19(1):17–37.

Friedman-Hill, E. 2005. Jess, the rule engine for the java platform. Sandia National Laboratories.

Gu, T.; Wang, X. H.; Pung, H. K.; and Zhang, D. Q. 2004. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 270–275.

Hanson, J. 2005. Event-driven services in soa. JavaWorld.

He, H. 2003. What is service-oriented architecture? O'Reilly Web services.XML.com.

H.Wang, X.; Gu, T.; Zhang, D. Q.; and Pung, H. K. 2004. Ontology based context modeling and reasoning using owl. In *Proceedings of Workshop on Context Modeling and Reasoning(CoMoRea'04)*, 18–22.

Jih, W. R.; Cheng, S. Y.; and Hsu, J. Y. J. 2005. Context-aware access control on pervasive healthcare. In *EEE'05 Workshop: Mobility, Agents, and Mobile Services (MAM)*.

Schulte, R. 2004. Event-driven architecture: The next big thing. Application Integration and Web Services Summit. Gartner, Inc.